

Программирование

Введение

Кафедра ИВТ и ПМ

2018



План

Программирование

О Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

- Литералы

- Переменные

- Операции

- Определение типа

Ввод и вывод



Outline

Программирование

О Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод



Программирование

Программирование — процесс создания компьютерных программ.



Программирование

Программирование — процесс создания компьютерных программ.

Программирование = алгоритмы + структуры данных.

—Н. Вирт



Структура данных - программная единица, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных данных.



Алгоритмы и структуру данных

Структура данных - программная единица, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных данных.

Алгоритм — набор инструкций, описывающих порядок действий исполнителя для достижения некоторого результата.



Основные способы записи алгоритмов:

- ▶ **вербальный**, когда алгоритм описывается на человеческом языке;
- ▶ **графический**, когда алгоритм описывается с помощью набора графических изображений;
- ▶ **символьный**, когда алгоритм описывается с помощью набора символов (на языке программирования, ЯП).



Вербальное представление алгоритма. Пример

В случае термического ожога глаз нужно:

1. Срочно изолировать больного от яркого света.
2. Закапать глаза 0,5% раствором дикаина, лидокаина или новокаина.
3. Провести внутреннее обезболивание (прием анальгетика).
4. Закапать глаза 30% раствором сульфацил-натрия или 2% раствором левомицетина.
5. Немедленно следовать в больницу.



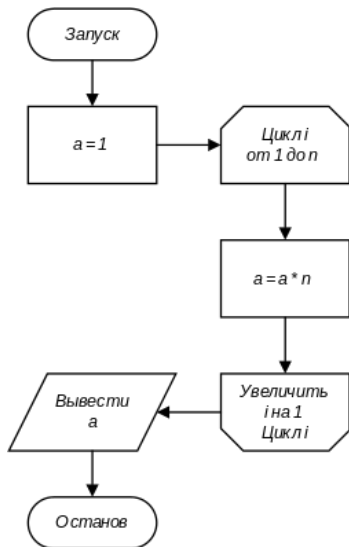
Символьная запись алгоритма. Пример

```
print("Квадраты чисел")  
  
nums = [1, 2, 7, 13, 17]  
  
for i in nums:  
    print( i, " ", i**2 )
```



Графическое представление алгоритма. Пример

Блок-схема



Классификация языков программирования

Языки низкого и высокого уровня

- ▶ **Низкоуровневый язык программирования** (язык программирования низкого уровня) — язык программирования, близкий к программированию непосредственно в машинных кодах (которые исполняются процессором).
- ▶ **Высокоуровневый язык программирования** — язык программирования, разработанный для быстроты и удобства использования программистом.



Машинный код

Программа «Hello, world!» для процессора архитектуры x86
(ОС MS DOS)

```
BB 11 01 B9 0D 00 B4 0E 8A 07 43 CD 10 E2 F9 CD 20 48 65 6C  
6C 6F 2C 20 57 6F 72 6C 64 21
```



Код низкоуровневого языка

Программа «Hello, world!» на языке ассемблера

```
mov bx, 0111h  
mov cx, 000Dh  
mov ah, 0Eh  
mov al, [bx]  
inc bx  
int 10h  
loop 0108  
int 20h  
HW db 'Hello, World!'
```



Код языка высокого уровня

Программа «Hello, world!» на языке программирования Python

```
print( "Hello, world!" )
```



Классификация языков программирования

Компилируемые и интерпретируемые языки

Компиляция — трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду (абсолютный код, объектный модуль, иногда на язык ассемблера)

Интерпретация — процесс одновременного чтения и выполнения исходного кода.



Язык программирования

Язык программирования задается тремя компонентами:

- ▶ алфавитом,
- ▶ синтаксисом
- ▶ семантикой.



Язык программирования

Алфавит - это набор различных символов: букв, цифр, специальных знаков и т. п.

Например, алфавит машинного языка состоит из двух символов: 0 и 1, а если программа записана в восьмеричной системе счисления, то из восьми символов: 0, 1, 2, 3, 4, 5, 6 и 7.

Синтаксис языка программирования — набор правил, описывающий комбинации символов алфавита, считающиеся правильно структурированной программой (документом) или её фрагментом

Семантика - это смысл синтаксических конструкций.



Outline

Программирование

О Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

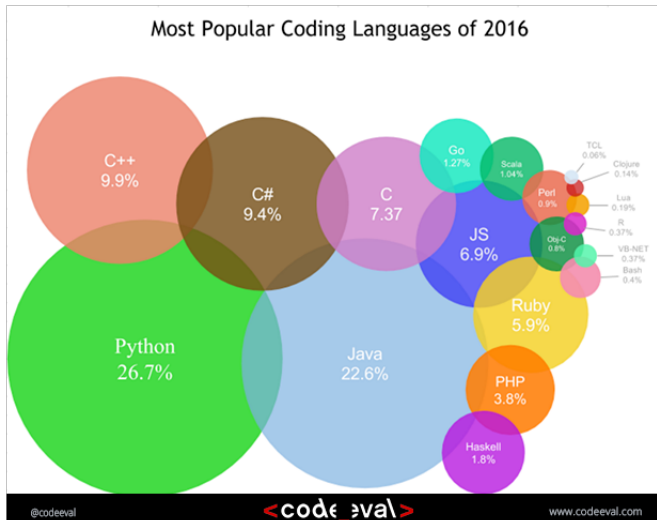
Операции

Определение типа

Ввод и вывод



Most Popular Coding Languages of 2016



Кто использует?

Google



CANONICAL

NETFLIX

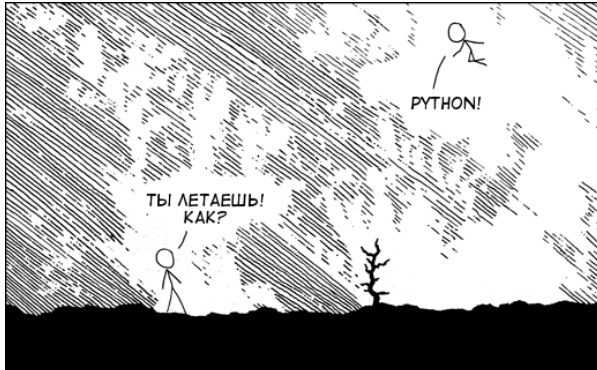


YAHOO!



Organizations Using Python

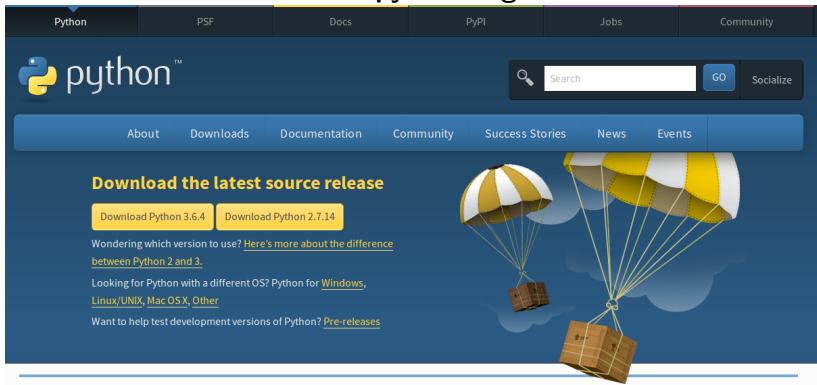




Где скачать?

Интерпретатор и базовые средства работы с ним

www.python.org



Во многих ОС семейства Linux (например **Ubuntu**)
интерпретатор Python уже установлен.



Outline

Программирование

О Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод



Реализация языка программирования

- ▶ Транслятор (компилятор или интерпретатор)
- ▶ Отладчик
- ▶ Набор библиотек
- ▶ Документация

Integrated Development Environment (IDE) = Транслятор +
Отладчик + редактор исходных кодов + документация



IDE - интегрированные среды разработки

- ▶ Wing

Простая

- ▶ Jupiter (Anaconda)

Хорошо подходит для небольших программ и экспериментов, принцип работы поход на matlab, нет отладчика.

- ▶ PyCharm

Удобный, функциональный, тяжёловесный.

- ▶ ...



О Python

- ▶ Высокоуровневый язык
- ▶ Общего назначения
- ▶ Ориентирован на повышение производительности разработчика
- ▶ Интерпретируемый
- ▶ Компилируется в байт-код
- ▶ Объектно ориентированное программирование
- ▶ Функциональное программирование
- ▶ Рефлексивность
- ▶ Динамическая типизация
- ▶ Сборка мусора



Философия

import this

- ▶ Простое лучше, чем сложное.
- ▶ Сложное лучше, чем запутанное.
- ▶ Плоское лучше, чем вложенное.
- ▶ Разреженное лучше, чем плотное.
- ▶ Читаемость имеет значение.
- ▶ Встретив двусмысленность, отбрось искушение угадать.
- ▶ Должен существовать один — и, желательно, только один — очевидный способ сделать это.
- ▶ Сейчас лучше, чем никогда.
- ▶ Если реализацию сложно объяснить — идея плоха.
- ▶ Если реализацию легко объяснить — идея, возможно, хороша.
- ▶ ...



Как работать с Python?

- ▶ В интерактивном режиме
Каждая команда (может быть составной) выполняется тут же
- ▶ Программы в отдельном файле

Распространены две версии языка: 3.x и 2.x.

Версия 2.x до сих пор используется только из-за того, что на ней было написано много кода на момент появления 3-й версии, которая внесла существенные изменения в язык.



Outline

Программирование

О Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод



Интерактивный режим

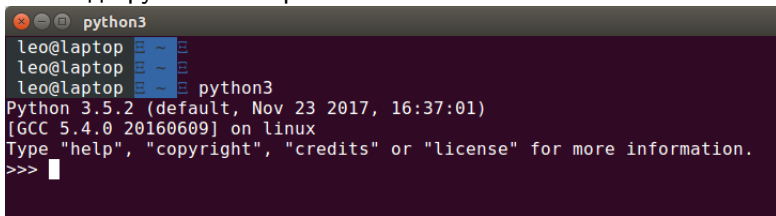
Запуск интерпретатора Python в интерактивном режиме:

- ▶ Windows

Запустить Python непосредственно или `python.exe` (из командной строки)

- ▶ Linux

Команда `python3` в терминале



```
python3
leo@laptop ~$
leo@laptop ~$
leo@laptop ~$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```



Интерактивный режим интерпретатора

Использование python как калькулятора в интерактивном режиме

- ▶ работает как командная строка
- ▶ умеет производить математические вычисления
- ▶ результат вычисления сразу отображается на экране
- ▶ результат вычисления автоматически (неявно) записывается в переменную `_` (знак подчёркивания)
- ▶ чтобы просмотреть содержимое переменной достаточно указать её имя (идентификатор)

```
>>> имяпеременной
```



Интерактивный режим

Использование python в качестве калькулятора

- ▶ Доступные арифметические операции: +, -, *, /
- ▶ Возведение в степень, x^y

```
3.14**3
```

```
=30.959144000000002
```

- ▶ Целочисленное деление

```
7 // 3
```

```
= 2
```

- ▶ Остаток от деления

```
12 % 5
```

```
= 2
```



Outline

Программирование

О Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод



Как сделать больше, чем простые математические вычисления?

Компилятор или интерпретатор языка программирования распространяются вместе со набором подпрограмм (функций), готовых для использования программистом.

Этот набор подпрограмм, типов данных, переменных и т.п. называется **стандартной библиотекой**.



- ▶ Стандартная библиотека разделена на части - **модули**.
- ▶ Каждый модуль содержит подпрограммы и другие элементы предназначенные для определённой цели. Например для математических вычислений или для работы с изображениями.
- ▶ Некоторые подпрограммы являются частью языка (а не стандартной библиотеки). Например подпрограммы вывода данных на экран **print** и ввода данных с клавиатуры **input**.



Модули

- ▶ Модули позволяют программисту воспользоваться уже готовыми подпрограммами, чтобы уже с их использованием строить свои программы.
- ▶ Чтобы воспользоваться подпрограммами модуля, для начала его нужно *подключить*.
- ▶ Для подключения модуля используется оператор (команда) **import**, следом за которой через пробел указывается имя модуля.

```
import module_name
```



Подключение модулей

чтобы использовать математические функции и константы
нужно подключить модуль `math`

```
import math
```



Использование модулей

Чтобы вызывать функцию из этого модуля следует указать его имя и уже потом, через точку, имя функции
Например:

```
import math
```

```
math.sin ( math.pi / 2 )  # 1.0  
math.degrees( pi / 2 )    # 90. радианы -> градусы.  
math.radians( 90 )        # 3.14. градусы -> радианы  
math.exp ( 2 )            #  $e^2 = 7.39$   
math.sqrt(81)             # 9
```

решётка - знак комментария.



Использование модулей

Однако каждый раз указывать имя пакета неудобно. Поэтому можно при его подключении указать конкретные функции, что будут использованы:

```
from math import tan, sin
```

Теперь можно использовать функции `tan` и `sin` непосредственно, без указания имени пакета:

```
sin (3.1415) ##
```

```
tan (3.1415 / 4) ##
```



Использование модулей

Если функций много, то их перечислять необязательно. Вместо этого можно сделать доступным всё содержимое пакета:

```
from math import *
```

Теперь можно использовать все функции пакета непосредственно



Демонстрация



Outline

Программирование

О Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод



Алфавит языка Python

- ▶ Буквы национальных алфавитов, А-з, А-я, ...
- ▶ Символ подчёркивания `_`
- ▶ Цифры 0-9
- ▶ Специальные символы `+ - * / > < = ; ' , . : [] () @`
- ▶ Комбинации символов (считаются одним символом)



Лексема (token) - последовательность допустимых символов языка программирования, имеющая смысл для транслятора.

Виды лексем

- ▶ Идентификаторы (имена, identifiers)
- ▶ Служебные (зарезервированные слова, keywords)
- ▶ Литералы (неименованные константы, literals)
- ▶ Знаки операций (punctuators)



Идентификаторы (имена, identifiers)

Идентификатор — это имя программного объекта: переменной, константы, массива, функции, класса и т. п.

Ограничения идентификаторов

- ▶ первый символ не должен быть числовым символом;
- ▶ первый символ может быть любой алфавитный символ (в том числе любой UTF-8 символ национальных алфавитов);
- ▶ далее в имени можно использовать как алфавитные, так и числовые символы, за исключением пробельных символов;
- ▶ в качестве имени нельзя использовать служебные (ключевые) слова;
- ▶ в python регистро-зависимые имена, т. е.: BookId, bookID, Bookid, bookid, bookId и т. д., — разные имена.



Служебные слова (keywords)

```
False    class    finally is    return  
None     continue for lambda try  
True     def from  nonlocal  while  
and del global not with  
as elif  if or yield  
assert else import pass  
break except in raise
```



Outline

Программирование

О Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод



Литералы

Литерал (Literal) — запись в исходном коде компьютерной программы, представляющая собой фиксированное значение.

Литералы классифицируются по типам значений.



Литералы

► Литералы вещественного типа

1.23

1.

.123

3.14e-10 # *3.14 ^-10*

4E210 # *4^210*

4.0e+210 # *4^210*



Литералы

- ▶ Литералы комплексного типа

$3 + 4j$ *# 3 + 4i*

$3.0 + 4.0j$ *# то же самое*

можно писать большую J

$3J$ *# 0 + 3i.*

$2 + 0j$ *#*

Для обозначения мнимой части комплексного числа используют постфикс *j*.



Литералы

- ▶ Литералы логического типа

True

False

Регистр имеет значение.



Литералы

► Строковые литералы

`'qwerty'` # можно использовать одинарные кавычки

`"abcdef"` # а можно двойные

в двойных кавычках одинарные считаются отдельным

`"Can't"`

и наоборот

`'ФГБОУ ВО "ЗаБГУ"'`

`"""текст в тройных кавычках может располагаться
на нескольких строках.`

`ещё одна строка`

`и ещё"""`

Неправильно:

`'кавычки должны соответствовать друг другу'`



Литералы

- ▶ Пустое значение

`None`



Тип данных (тип):

- ▶ множество допустимых значений, которые могут принимать данные, принадлежащие к этому типу;
- ▶ набор операций, которые можно осуществлять над данными, принадлежащими к этому типу.



Типы

Простые типы

- ▶ NoneType
- ▶ bool
- ▶ int
- ▶ float

Составные типы

- ▶ str - строка
- ▶ tuple - кортеж
- ▶ list - список
- ▶ map - отображение (ассоциативный массив)



Outline

Программирование

О Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод



Переменные

Переменная (в Python) - имя, с которым может быть связано значение.



Объявление (declaration) - указание идентификатора (и его свойств). Объявление используется, чтобы уведомить компилятор о существовании элемента.

Определение (definition) - одновременное указание идентификатора и его значения. Объявление используется, чтобы уведомить компилятор о существовании элемента и одновременно задать значение.



В Python Нельзя объявить переменную не задав ей значения

Т.е. идентификатор любой переменной должен быть определён.

Если переменную объявить нужно, но пока не ясно какое значение ей задать можно использовать пустое значение None:

```
a = None
```



Переменные

Переменные также как и литералы классифицируются по типам записанных в них значений.

Тип переменной определяется типом записанного в неё значения.

```
a = 42      # переменная целого типа
x = 3.14    # ... вещественного типа
s = "Hello, Python!" # ... строкового типа
b = True    # ... логического типа
```

Регистр имеет значение: А и а - разные переменные.

Символ = оператор присваивания, он связывает имя переменной и значение ("записывает значение в переменную")



Переменные

Определим типы переменных (см. предыдущий слайд) с помощью функции `type`

```
type(a)    # int  
type(x)    # float  
type(s)    # str  
type(b)    # bool
```



Outline

Программирование

О Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод



Операции с вещественным типом

- ▶ Арифметические
+, -, *, // (целочисленное деление), % (остаток от деления), **
- ▶ Сравнение
== (равно), != (не равно), <, >, <=, >=
- ▶ Принадлежность диапазону

$x < y < z$

то же самое что и

$x < y$ and $y < z$



Операции с целым типом

- ▶ те же самые, что и с вещественным типом
- ▶ побитовые операции
 - » > " (побитовый сдвиг вправо)
 - < < (побитовый сдвиг влево)



Outline

Программирование

О Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод



Типы и преобразование типов

Какие типы будут у этих переменных?

```
x = 300 + 2.0
```

```
y = 12 + 8 // 2
```

```
z = sin( pi ) + 1
```

```
a = 12.0 // 4
```

```
b = 12.0 % 5
```



Преобразование типов

Какие типы будут у этих переменных?

```
x = 300 + 2.0      # float
y = 12 + 8 // 2     # int
z = sin( pi ) + 1   # float
a = 12.0 // 4       # float
b = 12.0 % 5        # float
```



Преобразование типов

Тип определяется по типу литерала.

Тип всегда приводится к наиболее общему.

Тип `float` является более общим чем `int`.

Тип `complex` является более общим для `float`.

Например при сложение вещественного и целого числа результат будет вещественного типа



Явное преобразование типов

`int(выражение)`

преобразует выражение в целое число

```
int("123")    # строка -> целое, 123
int(5.125)    # вещественное число -> целое.
               # дробная часть отбрасывается
```

```
float("123")  # строка -> вещесств. число, 123.0
float(123)    # 123 -> 123.0
```

```
str(123)      # 123 -> "123"
str(5.125)    # 5.125 -> "5.125"
```



Outline

Программирование

О Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод



Вывод данных

```
print( ... )  
  
print ( "Hello, Python!" )  
  
a = 10  
b = 3.14  
print("a = ", a, "; b = ", b)  
# будет напечатано:  
# a = 10; b = 3.14
```

Вывести на экран и не переходить на следующую строку:

```
print ( "Hello, Python!", end="")
```

[pythoner.name/formatted-output](#) - форматирование вывода



Ввод данных с клавиатуры

Функция **input** ожидает ввод с клавиатуры и возвращает значение введённое пользователем.

```
s = input()
```

в переменную s будут записаны введённые данные

Это значение имеет строковый тип (str), вне зависимости от того, что ввёл пользователь.

По окончании ввода пользователь должен нажать Enter.



Ввод данных с клавиатуры

Ввод пользователя необходимо преобразовывать к требуемому типу данных.

Например, чтобы преобразовать строку в целое и вещественное число нужно использовать явное преобразование типа:

```
d = int( input() )
```

```
f = float( input() ) # десятичный разделитель - точка
```

В этих примерах выполнение функции начинается с самого нижнего уровня вложенности: сначала `input`, затем результат выполнения функции `input` будет помещён в функцию `int`, а она уже преобразует его в значение целого типа.



Ввод данных с клавиатуры

Введённые данные при преобразовании должны быть корректными.

Например, невозможно преобразовать строки "data" или "три" в число с помощью функций `int` или `float`.

Если не удаётся преобразовать строку в число, то программа завершится аварийно с одной из следующих ошибок:

```
ValueError: invalid literal for int() with base 10  
ValueError: could not convert string to float
```



Ввод данных с клавиатуры

Нужно однозначно сообщать пользователю смысл и формат входных данных, чтобы в большинстве случаев избежать ошибок при преобразовании значений.

При этом пояснения должны быть лаконичны.

```
print("Введите ваш вес в кг")
m = float( input() )
print("Введите ваш рост в метрах")
h = float( input() )

print("Индекс массы тела равен ")
print( m / h**2 )
```



Ссылки и литература

ЭБС

- ▶ biblio-online.ru - ЭБС Юрайт
- ▶ studentlibrary.ru - ЭБС "КОНСУЛЬТАНТ СТУДЕНТА"
- ▶ Федоров, Д. Ю. Программирование на языке высокого уровня python : учебное пособие для прикладного бакалавриата Содержит краткое описание языка.
- ▶ ru.wikibooks.org/wiki/Python - Викиучебник
- ▶ Лутц М. Изучаем Python. 2010. - 1280 с. Содержит подробное описание языка.
- ▶ Официальная документация Python3
`help(имя)`



Дополнительная литература

- ▶ O'Connor T.J. Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers. 2012 — 288 p.
- ▶ Numerical methods in engineering with Python 3 / Jaan Kiusalaas.



Ссылки и литература

Ссылка на слайды

github.com/VetrovSV/Programming

